3. Analízis modell kidolgozása

35 – negalt

Konzulens:
Ludmány Balázs

Csapattagok

Gutai Auguszta Matild	QWG59W	auguszta123@gmail.com
Fábián Csenge Zita	KGM10V	csengefabian@gmail.com
Ilosvay Viktória Brigitta	M27F3W	ilosviki@gmail.com
Szabó Marcell	LAUPQQ	szabomarcell.usa@gmail.com
Szabó Kinga Johanna	G003M6	szabo.kinga1999@gmail.com

2020-02-26

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Ásó

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy használata során egy réteg hó helyett két rétegnyi havat tudjanak ellapátolni ugyanannyi munka ráfordítása mellett.

3.1.2 Búvárruha

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy ha később vízbe esnének, minden gond nélkül ki tudjanak menekülni maguktól.

3.1.3 Eszkimó

A jatékosok egyik fajtája, speciális képessége az igloo építés. Az eszkimó testhője maximum 5 egységnyi lehet.

3.1.4 Étel

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy elfogyasztása során növelhessék testhőjüket.

3.1.5 Iglu

Az iglu egy jégmezőn elhelyezhető építmény. Megépítéséhez csak az eszkimó ért, de ezt követően minden mezőn tartózkodó játékos számára védelmet nyújt a vihar elől.

3.1.6 Játékos

A Játékosokat foglalja össze, akik lehetnek eszkimók vagy sarkkutatók. Ők a játék aktív résztvevői, a pályán mezőről-mezőre mozognak, körönként 4 egység munkát végezve próbálják elérni a csapat együttes célját, a jelzőrakéta összerelését és elsütését.

3.1.7 Jelzőfény

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

3.1.8 Jelzőrakéta

A patronból, pisztolyból és jelzőfényből összeszerelt tárgy, melyet a szereplőknek kulcsfontosságú feladatuk a játék megnyerésének célából összerakni és elsütni.

3.1.9 Jégmező

A mezőkből épül fel a játéktábla, a játékosok ezen mozoghatnak. A mezőkön lehet hó, és ennek szintje a viharok hatására növekedhet. A hó alatt a jégbe fagyva esetleg tárgyakat rejthet a mező. Egyes mezők teherbírása véges, ezekre nem léphet rá akárhány játékos, mert akkor felfordul.

3.1.10 Jégtábla (Játéktábla)

A mezők összességét szimbolizálja. A pálya mérete a játékosok számától függően különböző nagyságú lehet. A pálya négyzet alakú és tenger határolja, rajta szintén négyzet alakú mezők vannak. Ezek lehetnek jégmezők (stabil vagy instabil mezők) illetve lyukak.

3.1.11 Kötél

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki, hogy ha később egyik játékostársuk egy szomszédos mezőn vízbe esik, akkor ki tudják menekíteni. Mivel egy csapatként dolgoznak, fontos, hogy társaikat is megmentsék, különben ők sem fognak megmenekülni a zord sarki mezőről.

3.1.12 Lyuk

A jégtáblán lehetnek hóval fedett lyukak is. Ide lépve a játékos beleesik és megfullad, azaz vége a játéknak.

3.1.13 Patron

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

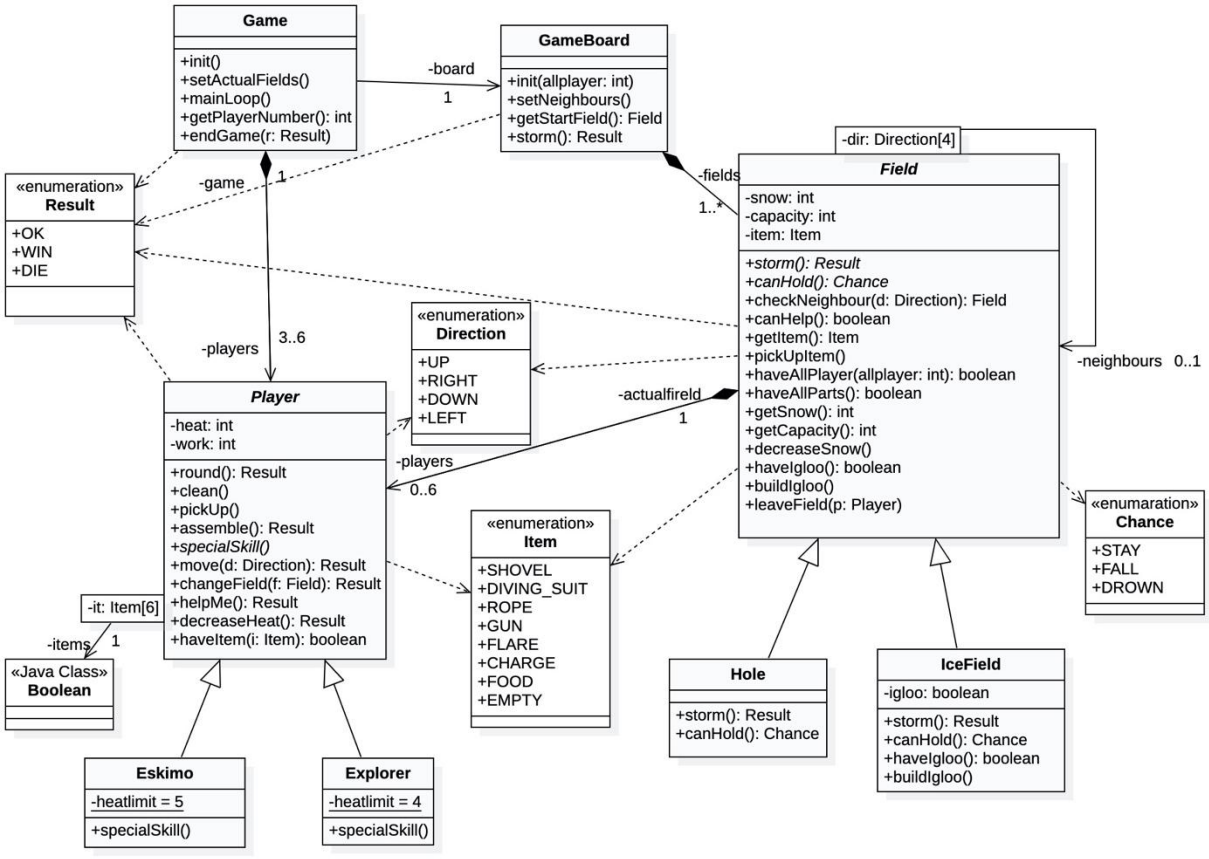
3.1.14 Pisztoly

A jégmezőkbe fagyva található tárgy, melyet a játékosok ásnak ki. Megtalálása nagyon fontos, mivel ez egy nélkülözhetetlen alkatrész a jelzőrakéta összeszereléséhez, amivel meg tudnak menekülni.

3.1.15 Sarkkutató

A játékosok másik fajtája, akik speciális képessége a jégtáblák teherbírásának kiderítése. A sarkkutatók testhője maximum 4 egységnyi lehet.

3.2 Statikus struktúra diagramok



Az <<instantiate>> típusú dependencyket az osztálydiagram nem tartalmazza, mert azok nagyon átláthatatlanná tennék az ábrát.
A Java Boolean osztálya azért szerepel az ábrán, mert a Player osztály Map struktúráját csak így lehetett megjeleníteni a diagramban (más megjelenítést nem engedélyezett a StarUML program)

3.3 Osztályok leírása

3.3.1 Chance

Felelősség

Enumeration osztály, amely megadja, hogy egy mezőre való lépés esetén hogyan reagál a jégtábla, és ez hogyan hat ki a játékosra.

Literálok

- **STAY**: Mezőre lépés esetén nem fordul át a jégtábla, tehát a maximum létszámot nem haladja meg az adott mezőn álló játékosok száma. Összegezve a játékos(ok) marad(nak) a jégtáblán.
- **FALL**: Lyukra lépés esetén a játékos beleesik a vízbe.
- **DROWN**: Instabil jégtáblára való lépés esetén a létszám meghaladás következtében átfordul a jégtábla, és a rajta álló játékosok a jég alá szorulva megfulladnak.

3.3.2 Direction

Felelősség

Enumeration osztály, amely a négy irány közül vehet fel egy értéket.

Literálok

- **UP**: Fel
- **RIGHT**: Jobb
- **DOWN**: Le
- **LEFT**: Bal

3.3.3 Eskimo

Felelősség

Eszkimó karaktertípus esetén megadja a maximális testhő mértékét, illetve kezeli az eszkimó különleges képességét, tehát az igloo építésének menetét.

Ősosztályok

- Player → Eskimo

Attribútumok

- **int heatlimit = 5**: Statikus attribútum. Az eszkimó testhő szintjének maximális számát adja meg.

Metódusok

- **void specialSkill()**: A Player osztályban levő absztrakt függvény megvalósítása. Legelőször is meghívja a haveIgloo() metódust. Amennyiben FALSE visszatérési értéke lesz, akkor semmi sem történik. Ha viszont ez TRUE, akkor a work attribútum értékének eggyel való csökkentése mellett meghívja a buildIgloo() függvényt.

3.3.4 Explorer

Felelősség

Sarkkutató karaktertípus esetén megadja a maximális testhő mértékét, illetve kezeli a sarkkutató különleges képességét, tehát egy szomszédos mező teherbírásának vizsgálatát.

Ősosztályok

- Player →Explorer

Attribútumok

- **int heatlimit = 4:** Statikus attribútum. A sarkkutató testhő szintjének maximális számát adja meg.

Metódusok

- **void specialSkill()**: A Player osztályban lévő absztrakt függvény megvalósítása. Először bekér egy irányt, majd erre meghívja a checkNeighbour függvényt. Amennyiben ez NULL értékkel tér vissza, akkor a játékosnak újra meg kell adnia egy irányt. Ellenben ha ennek nem NULL visszatérési értéke lesz, akkor a visszakapott mezőre meghívja a getCapacity() függvényt a work attribútum értékének eggyel való csökkentése mellett.

3.3.5 *Field*

Felelősség

Mezők kezelésére szolgáló absztrakt osztály. Segítségével vagy a játékosok, vagy a vihar által a mezőn végzett tevékenységeket lehet megvalósítani.

Ősosztályok

- Nincsenek.

Attribútumok

- **int snow:** Az adott mezőn lévő hóréteg mennyiségét tárolja.
- **int capacity:** Az adott mező teherbíró képességét tárolja el. Stabil jégtábla esetén a max játékosok száma, lyuk esetén pedig nulla az értéke.
- **Item item:** Tárolja, hogy a mezőn milyen tárgyat lehet felvenni. Ha nem található rajta, akkor EMPTY értéket ad vissza.
- **Map<Direction, Field> neighbours:** Tárolja a 4 irányban elhelyezkedő mezőt.
- **Player players:** Tárolja, hogy az adott mezőn melyik játékosok vannak rajta.

Metódusok

- ***abstract Result storm()***: Absztrakt függvény. A Hole vagy az IceField storm() függvénye kerül meghívásra.
- ***abstract Chance canHold()***: Absztrakt függvény. A Hole vagy az IceField canHold() függvénye hívódik meg.
- **Field checkNeighbour(Direction d)**: Az átadott irány alapján visszatér az abban az irányba levő objektum referenciájával. Ha arra tenger van, akkor ez az érték NULL lesz.
- **boolean canHelp()**: Akkor tér vissza TRUE értékkel, ha az adott mezőn tartózkodik menteni képes játékos. Ennek eldöntéséhez meghívja az összes rajta álló játékos haveItem(ROPE) függvényét. Ha legalább egy TRUE értékkel tér vissza, akkor ő is.

- **Item getItem()**: Visszatér a a mezőn található Item-mel.
- **void pickUpItem()**: EMPTY-re állítja az item attribútumát, mivel az addig tárolt Item-et egy játékos felvette.
- **boolean haveAllPlayer(int allplayer)**: Megvizsgálja, hogy az adott mezőn áll-e az összes játékos, tehát összehasonlítja a megkapott allplayer attribútumot a rajta állók számával. Ha ez megegyezik, akkor TRUE értékkel, ellenkező esetben FALSE értékkel tér vissza.
- **boolean haveAllParts()**: Megvizsgálja a haveItem(item) metódus segítségével, hogy az adott mezőn álló emberek együttesen rendelkeznek-e a három szükséges tárggyal (GUN, FLARE, CHARGE). Ha igen, akkor TRUE, máskülönben FALSE értékkel fog visszatérni.
- **int getSnow()**: Lekéri az adott mezőn lévő hórétegek számát.
- **int getCapacity()**: Visszaadja az mező teherbíró képességét. (Az elkészült játékban valószínűleg ez a függvény nem fog visszatérni a teherbírás értékével, hanem csak megjeleníti a képernyőn azt. Most a grafikus elemek hiányában illetve a jobb érthetőség kedvéért használjuk ezt a függvényt így.)
- **void decreaseSnow()**: Csökkenti az adott mezőn található hórétegek számát eggyel.
- **boolean haveIgloo()**: Megvizsgálja, hogy az adott mezőn található-e igloo. Ha van, akkor TRUE értékkel tér vissza.
- **void buildIgloo()**: Átállítja az igloo attribútum értékét FALSE-ról TRUE-ra.
- **void leaveField(Player p)**: Az attribútumban megkapott játékost kiveszi a players attribútumban tárolt játékosok referenciái közül (mivel a játékos elmozdult a mezőről).

3.3.6 Game

Felelősség

A játék elkezdésére/inicializálására, befejezésére, illetve a körök kezelésére szolgáló osztály. A viharok feltámadásnak valószínűségeit, illetve annak lebonyolításának kezdetét is ez az osztály kezeli.

Attribútumok

- **Player players**: Az adott játékosokat tárolására szolgáló tömb, melynek nagysága 3 és 6 között helyezkedik el (beleértve a határokat is).
- **GameBoard board**: A játékhoz tartozó játéktábla tárolására szolgál.

Metódusok

- **void init()**: A játék kezdetekor bekéri a játékosok számát majd sorra azoknak a karaktertípusát. Létrehozza a GameBoard-ot, majd meghívja az osztály init(player) metódusát átadva neki a játékosok számát. Ezt követően a bekért adatok alapján létrehozza az eszkimókat illetve a sarkkutatókat reprezentáló osztályokat. Végül meghívja a setActualFields() metódust.
- **void setActualFields()**: A korábban létrehozott Player példányoknak állítja be az actualfield attribútumát. Ehhez lekéri a kezdő mező referenciáját a GameBoard-tól a getStartField függvény segítségével.

- **void mainLoop()**: Ciklusban hívogatja meg a játékosok köreinek lezajlásáért felelős függvényeket. A ciklus minden lefutásakor először egy adott valószínűség alapján eldönti hogy jön-e vihar. Ha jön, akkor meghívja a Gameboard storm() függvényét (ellenkező esetben ez a függvényhívás kimarad). Ha ez nem DIE-al tért vissza, akkor meghívja a Player osztály round() metódusát. Amennyiben ez nem OK visszatérési értéket ad, vagy már korábban a storm() DIE-al tért vissza, akkor kilép a ciklusból. Ekkor meghívásra az endGamer(Result) függvény.
- **int getPlayerNumber()**: A players attribútumban tárolt játékosok számával tér vissza.
- **void endGame(Result r)**: A megkapott paraméter alapján eldönti, hogy hogyan végződik a játék, és befejezi azt.

3.3.7 GameBoard

Felelősség

Összefogja s egyben tárolja az összes mezőt. Emellett ha az adott körben úgy adta a gép, hogy lesz hóvihar, akkor kezeli, hogy az melyik mezőkre essen le.

Ősosztályok

- Nincsenek.

Attribútumok

- **Field fields**: A táblához tartozó mezők tárolására szolgál.

Metódusok

- **void init(int allplayer)**: Az átvett játékos-szám alapján létrehozza a mezőket majd eltárolja azokat fields tömbben. Végül meghívja a setNeighbours függvényt
- **void setNeighbours():** A már elkészített Field-eknek beállítja a szomszédait a fields tömb alapján.
- **Fied getStartField()**: Visszatér a bal felső sarokban lévő Field referenciájával. (Erről a mezőről fognak elindulni a játékosok).
- **Result storm()**: Eldönti egy adott valószínűség alapján minden egyes mezőre, hogy ott jön-e vihar. Ha jön, akkor meghívja annak a mezőnek a storm() függvényét. Futás végén, ha legalább egy mező storm függvénye DIE-al tért vissza, akkor ő is DIE-al tér vissza, különben OK-kal.

3.3.8 Hole

Felelősség

Lyukak kezelésére szolgáló osztály. A vele kapcsolatos kimentési kísérletet is ez az osztály kezdi meg.

Ősosztályok

- Field →Hole

Metódusok

- **Result storm()**: A Field osztályban levő absztrakt függvény megvalósítása. Az adott mező snow attribútumának értékét megnöveli eggyel. Mindig OK-kal tér vissza.
- **Chance canHold()**: A Field osztályban lévő absztrakt függvény megvalósítása. Minden esetben FALL értékkel tér vissza.

3.3.9 IceField

Felelősség

Jégtáblák/Jégmezők kezelésére szolgáló osztály. Egyrészt a jégmezők teherbírásának vizsgálatára, illetve az ezt követően megkapott adat alapján az esetek szétválasztására szolgál. Másrészt vihar esetén kezeli, hogy ha az adott mezőn esik a hó, akkor az milyen kritériumok mellett (van-e igloo vagy nincs) milyen következményekkel jár (mezőn levő hórétegek számát mindig növeljük, viszont a testhő csökkentése csak az iglooval védetlen mezőkön történik meg).

Ősosztályok

- Field →IceField

Attribútumok

- **boolean igloo**: Megadja, hogy adott mező tartalmaz-e igloot. (Ha tartalmaz, akkor TRUE az értéke)

Metódusok

- **Result storm()**: A Field osztályban levő absztrakt függvény megvalósítása. Az adott mező snow attribútumának értékét megnöveli eggyel. Amennyiben az adott mező nem tartalmaz igloo-t, akkor az ilyen mezőn álló játékosokra meghívja a decreaseHeat() metódust. Amivel ez a függvény visszatér, azzal tér vissza a storm() is.
- **Chance canHold()**: A Field osztályban levő absztrakt függvény megvalósítása. OK értékkel tér vissza, ha az adott mezőn levő játékosok számát még elbírja a jégtábla. Amennyiben már nem bírja el az adott létszámot, akkor DROWN értékkel tér vissza.
- **boolean haveIgloo()**:Ha van igloo a mezőn, akkor TRUE-val tér vissza, ellenkező esetben pedig FALSE-szal.
- **void buildIgloo()**: Akkor hívódik meg, ha egy eszkimó iglut épít a jégtáblán. Az igloo attribútumát TRUE értékre állítja.

3.3.10 Item

Felelősség

Enumeration osztály, amely a tárgyak közül vehet fel egy értéket. Célja, hogy megadjuk milyen elem van a jégbe fagyva. Ha semmi sincs, akkor az EMPTY érték lesz megadva. Ezen kívül a játékosnál lévő tárgyak azonosításához is erre az enumerációra van szükség. (A Map-ben ezek lesznek a minősítők.)

Literálok

- **CHARGE**: Patron
- **DIVING_SUIT**: Búvárruha
- **EMPTY**: Üres
- **FLARE**: Jelzőfény
- **FOOD**: Élelem
- **GUN**: Pisztoly
- **ROPE**: Kötél
- **SHOWEL**: Lapát

3.3.11 Player

Felelősség

Játékosok kezelésére szolgáló osztály. A játékosok munkájának és testhőjének vizsgálata mellett a körökkel és a bennünk elvégezhető cselekvésekkel foglalkozik. Minden játékos köre addig tart, amíg a work attribútumának értéke nem csökken le nullára. Ekkor egy újabb játékos körével fog foglalkozni. Minden cselekvés, ami az adott esetben engedélyezett, az egy egység munkavégzéssel jár (például tárgy felvétele olyan mezőn, amin még van hóréteg nem engedélyezett, és ilyenkor ez nem is jár munkavégzéssel).

Attribútumok

- **int heat**: Az adott játékos testhő szintjének mennyiségét tárolja.
- **int work**: Az adott játékos munkájának (szebben megfogalmazva: munkára fordítható energiájának) egységeit tárolja.
- **Map<Thing, boolean>**: Map struktúrában tárolja a játékosnál található tárgyakat. A Thing enumeráció elemei alkotják minősítőket, amelyekhez a struktúra boolean értékeket rendel, ezzel jelezve, hogy rendelkezik-e az adott játékos a tárgyal.

Metódusok

- **Result round()**: Elsőként beállítja a jelenlegi játékos work nevű attribútumának értéket négy egységre, majd végigvárja (egy ciklusban) a játékos lépéseit (míg a work értéke nulla nem lesz). Futás során az általunk választott cselekvésekhez szükséges függvényeket fogja meghívni. Ha az általa meghívott függvények OK-kal térnek vissza, akkor leellenőrzi a work értékét, és ha ez nem nulla, akkor folytatódik a ciklus futása.. Amennyiben ez az érték nullára csökken, akkor a round() metódus OK értéket ad vissza. Ha futás során bármely függvény DIE vagy WIN értékkel tér vissza, akkor a round kilép a ciklusából és ugyanazzal fog visszatérni.
- **void clean()**: A getSnow() függvény segítségével lekéri az adott mezőn levő hómennyiséget. Amennyiben ez nem nulla, akkor a decreaseSnow() metódus segítségével eggyel csökkenti a mező snow értékét, illetve a saját a work attribútumát is eggyel csökkenti. Ezután megvizsgálja az items nevű Map segítségével megvizsgálja, hogy a jelenlegi játékos rendelkezik-e ásóval (SHOWEL). Ha birtokol és van elég hóréteg, akkor mégegyszer meghívja a decreaseSnow() függvényt.
- **void pickUp()**: A getSnow() függvény segítségével lekéri az adott mezőn levő hómennyiséget. Amennyiben ez nulla, és a mező nem EMPTY vagy FOOD enumerációs értékkel rendelkezik, illetve ha nem rendelkezünk még az adott tárggyal (Map-ben FALSE az értéke), akkor a work értékének eggyel való csökkentése mellett felvesszük a tárgyat (M
- ap-ben TRUE-ra állítjuk át az értéket).
FOOD enumerációs érték esetén ellenőrizük, hogy a játékos heat attribútuma kisebb-e mint a heatlimit. Ha igen, akkor az aktuális játékos work attribútum eggyel való csökkentése mellett a heat attribútumot megnöveljük eggyel.
Sikeres tárgyfelvételkor meghívjuk a mező pickUp() függvényét.
- **void assemble()**: A work attribútumot eggyel csökkenti. Ezt követően a game attribútumban tárolt Game-re meghívja a getPlayerNumber() függvény. A visszakapot éréket átadja az általa hívott a haveAllPlayers() metódusnak. Ha ez FALSE értékkel tér vissza, akkor ő OK-kal fog. TRUE esetén viszont meghívja a HaveAllParts() függvényt, amely ha TRUE-val tér vissza, akkor az assemble() WIN-nel fog. Amennyiben a haveAllParts() metódus FALSE értékkel tér vissza, akkor az assemble() OK-kal fog.
- ***abstract void specialSkill()***: Absztrakt függvény. Az Eskimo vagy az Explorer osztály specialSkill() függvénye hívódik meg.

- **Result move(Direction d)**: A függvényben átadott irányra meghívja a checkNeighbour(Direction) metódust. Ha ez NULL-lal tér vissza, akkor semmi sem történik. Minden más esetben pedig a work attribútum értékének eggyel való csökkentése mellett visszatér a szomszédos mező referenciájával. Ezután meghívja a changeField(Field) függvény, aminek a megkapott referanciát adja át. Ami a changeField függvénynek a visszatérési értéke lesz, az lesz a move(Direction)-é is.
- **Result changeField(Field f)**: Meghívja az actualfieldben tárolt mezőre a leaveField függvényt, aminek önmagát adja át attribútumként. Majd beállítja a jelenlegi játékos actualfield nevű attribútumának értékét a megkapott mezőre. Ezt követően meghívja a Field osztály canHold() függvényét. Amennyiben ez DROWN-nal tér vissza, akkor a changeField metódus DIE-al fog. Ha FALL-al, akkor a meghívásra kerül a helpMe() függvény. Amilyen értékkel a helpMe() metódus visszatér, olyannal fog a changeField(Field) is. Végezetül ha a canHold függvény a STAY visszatérési értéket adja, akkor a changeField(Field) OK értékkel fog visszatérni.
- **Result helpMe()**: Először megvizsgálja a tárgyakat tároló Map-ben, hogy az adott játékos rendelkezik-e buvárruhával. Ha birtokol, akkor egy irány választását követően meghívódik erre az irányra a checkNeighbour(Field) függvény. Abban az esetben ha ez NULL értékkel tér vissza, akkor újra irányt kell választanunk addig, amíg jó irányt nem választ. Haez sikerült, akkor ezután meghívódik a changeField(Field) függvény, ami neki a megkapott fieldet kapja attribútumként. Ezt követően a helpMe()a meghívott changeField visszatérési értékével tér vissza.
Amennyiben, nincs a Map-ben buvárruha, akkor a heat attribútum értékét csökkenti eggyel. Ezután ellenőrzi, hogy a heat értéke lecsökkent-e nullára. Ha igen, akkor a helpMe() visszatérési értéke DIE lesz. Ellenkező esetben pedig a Field osztály checkNeighbour(Field) függvénye kerül meghívásra az actualfield attribútum alapján. (sorra fel-jobb-le-bal). Az erre kapott referenciákra hívódik meg sorra a Field osztály canHelp() metódusa. Az első TRUE értéket visszaadó mezőre átlép a beesett játékos (kimenekül a lyukból), tehát meghívódik a changeField(). Amilyen értékkel tér ez vissza, olyannal fog a helpMe() is. Ha a canHelp() függvények közül egyik se tér vissza TRUE-val, akkor a helpMe metódus DIE értékkel tér vissza.
- **Result decreaseHeat()**: Csökkenti a játékos heat nevű attribútumának értékét eggyel, majd megvizsgálja, hogy mennyi a heat értéke. Amennyiben ez nullára csökkent, akkor DIE, minden más esetben pedig OK visszatérési értéke lesz.
- **boolean haveItem(Item i)**: TRUE visszatérési értéket kapunk, amennyiben a Map-ben található ilyen tárgy. (Adott játékos rendelkezik vele)

3.3.12 Result

Felelősség

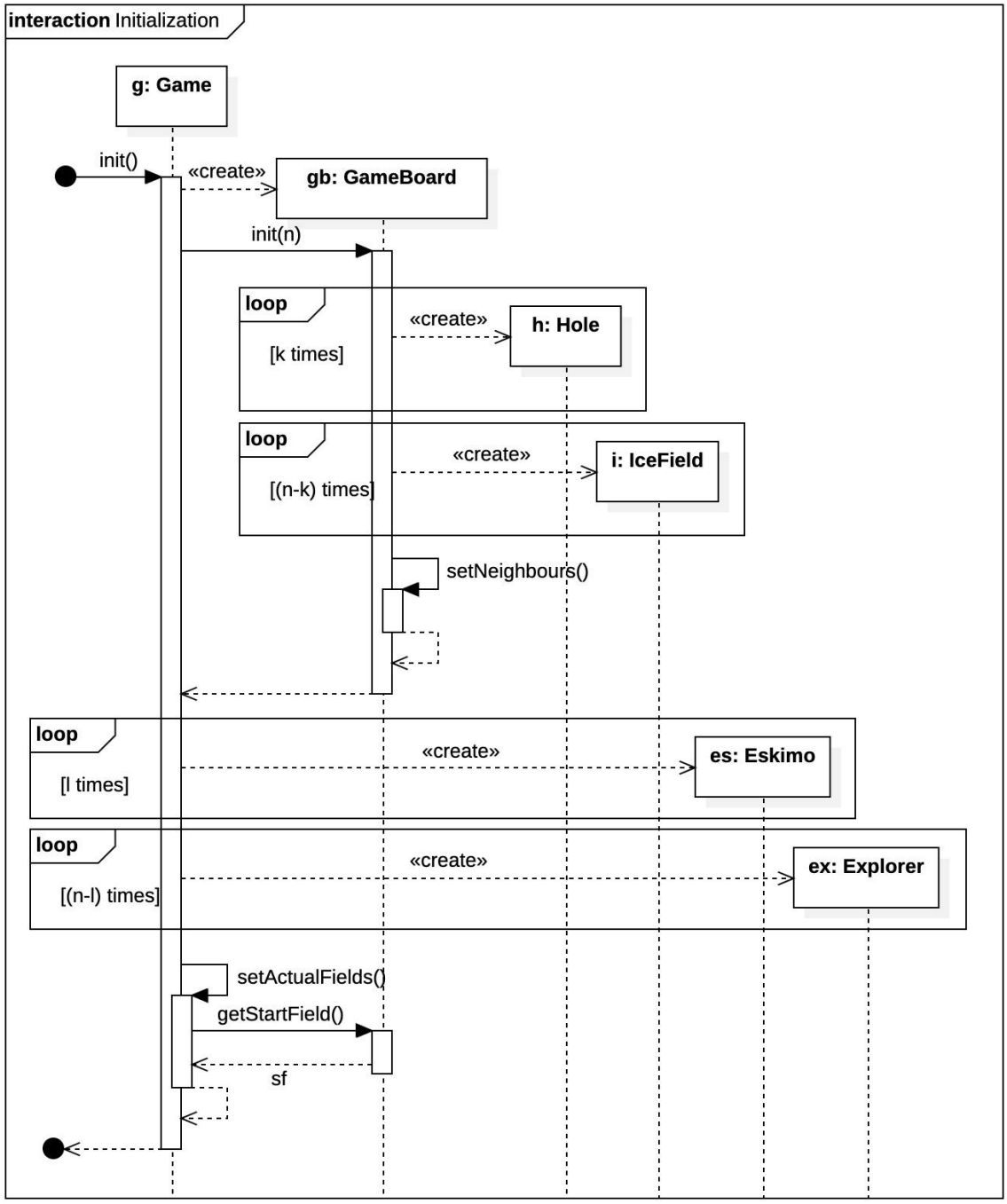
Enumeration osztály, amely a meghívott függvények futása alatt bekövetkező eseményeket reprezentálja. A legtöbb függvény visszatérési értéke ezen enumeráció értékei közül vesz fel egyet.

Literálok

- **WIN**: Győzelem (Jelzőpisztoly összeszerelése és elsütése)
- **DIE**: Halál miatt a játékosok elvesztették a játékot
- **OK**: Az előző elemek közül egyik se. (Se nem halt meg senki, illetve a csapat se nyerte meg a játékot)

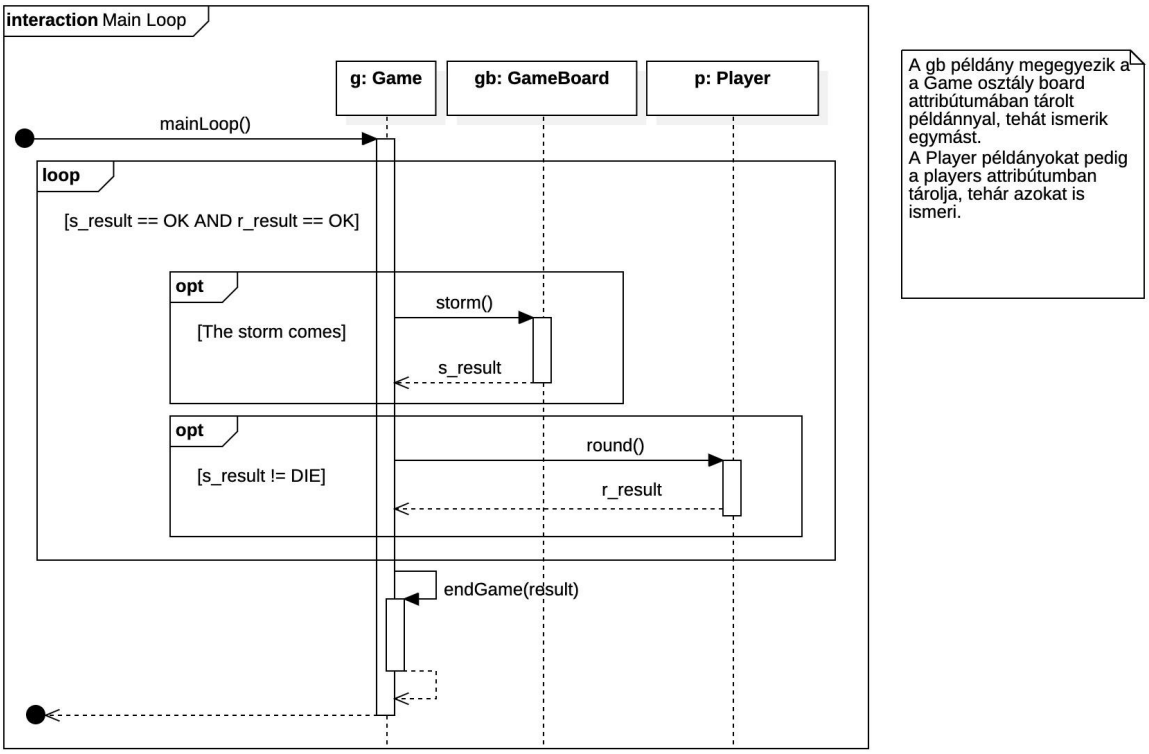
3.4 Szekvencia diagramok

3.4.1 Initialization



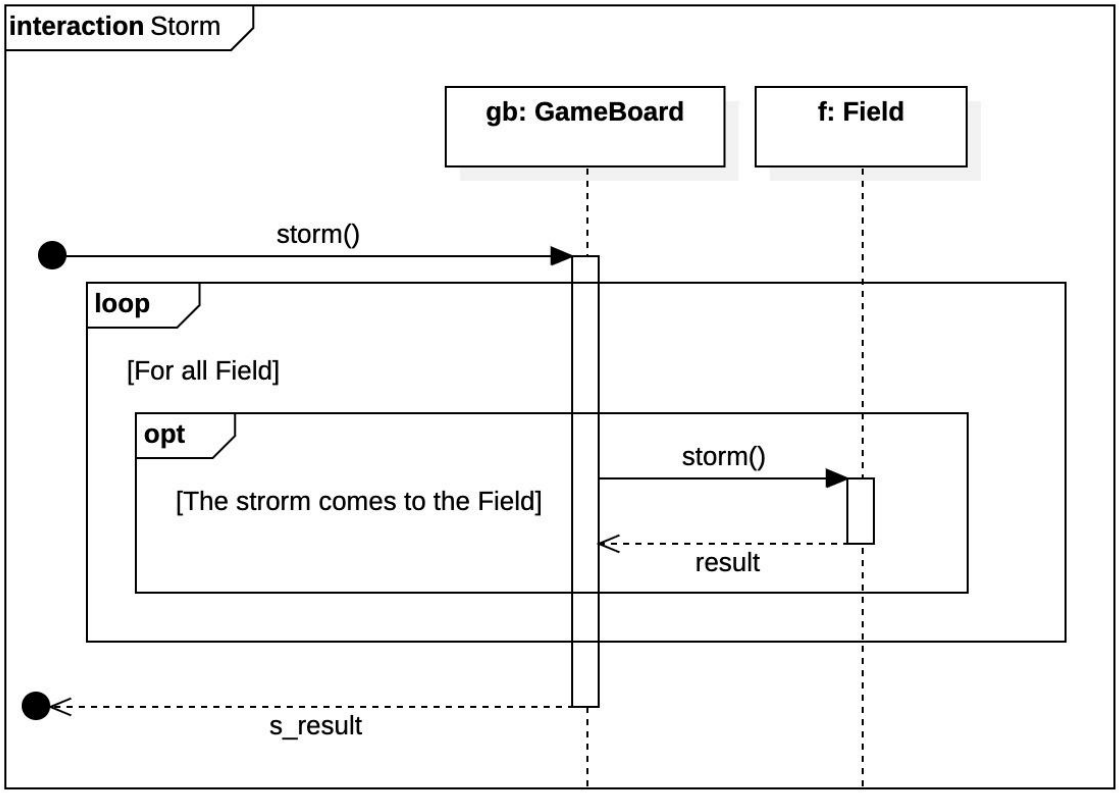
A gb példány megegyezik a Game osztály board attribútumában tárolt példánnyal, tehát ismerik egymást.
Az n a játékosok számát adja meg.

3.4.2 Main Loop



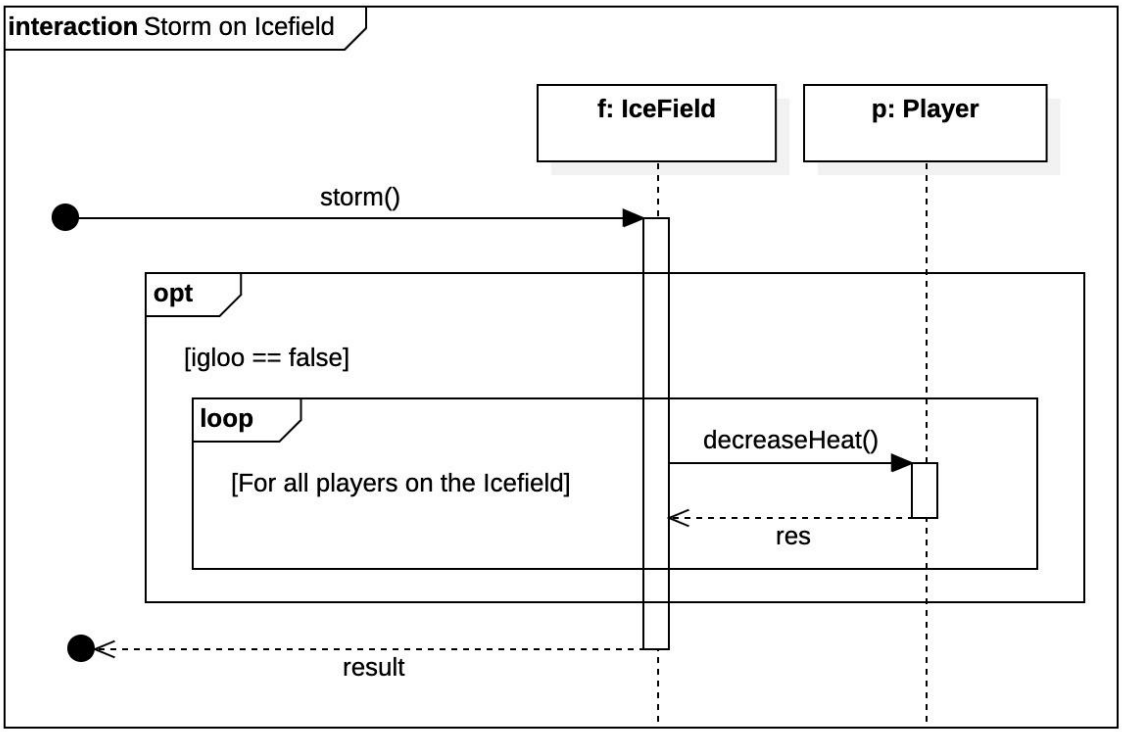
A gb példány megegyezik a a Game osztály board attribútumában tárolt példánnyal, tehát ismerik egymást.
A Player példányokat pedig a players attribútumban tárolja, tehár azokat is ismeri.

3.4.3 Storm



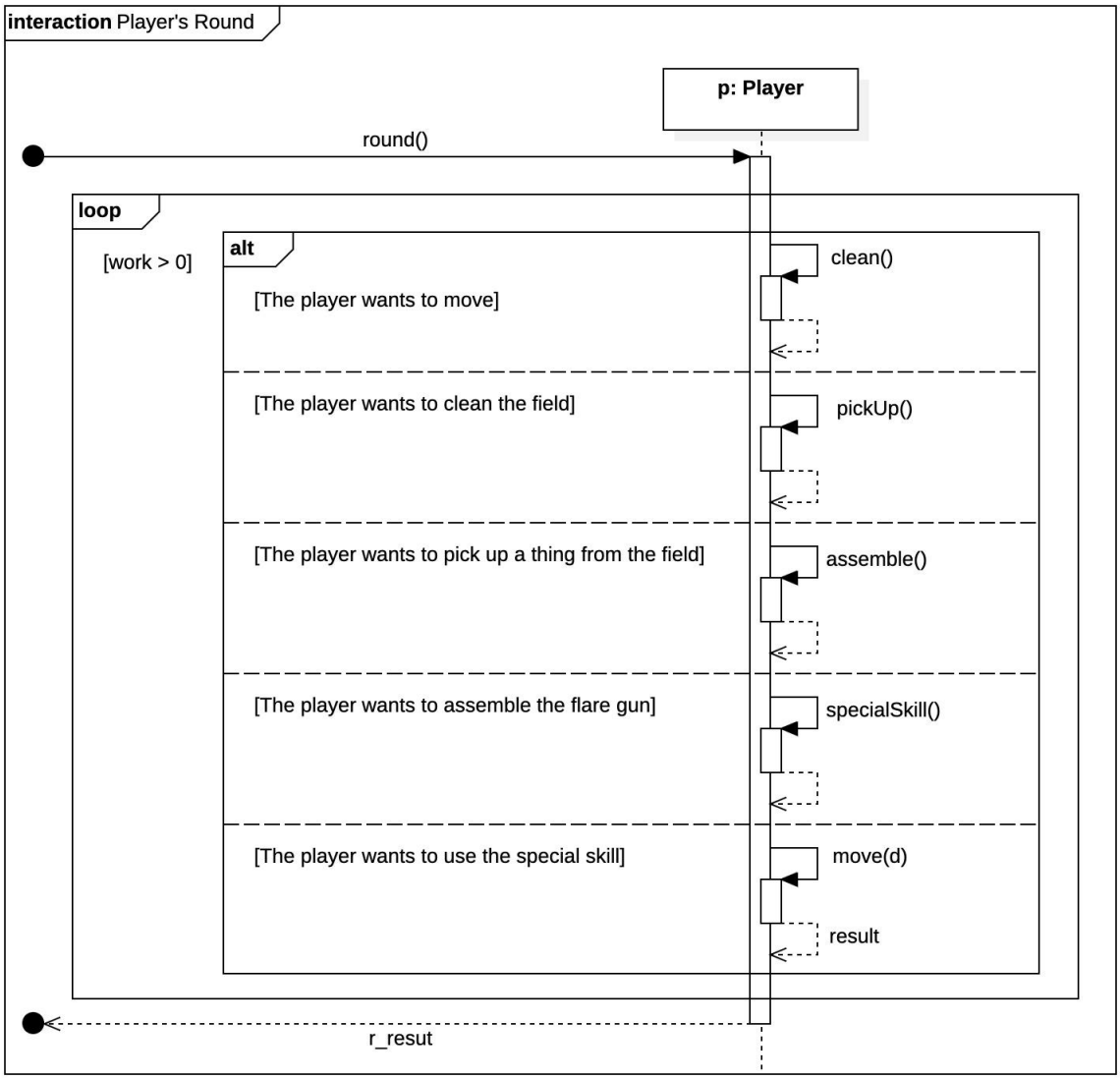
Az Field példányokat a GameBoard osztály a fields attribútumában tárolja, tehát ismerik azokat.

3.4.4 Storm on Icefield

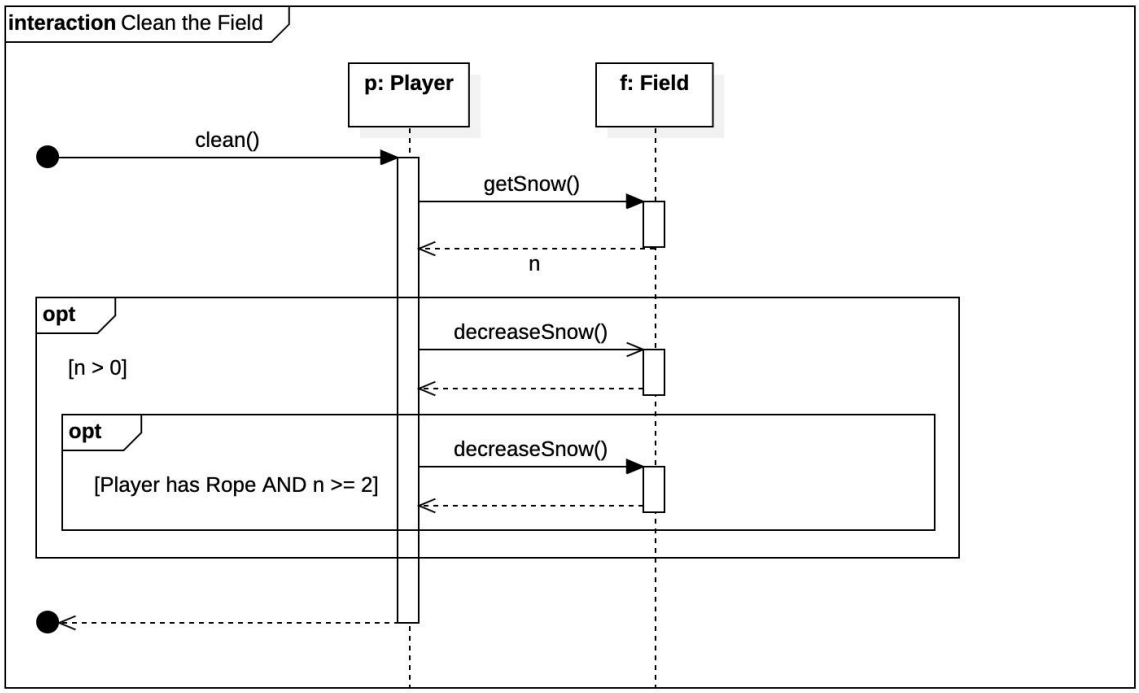


A Player példányokat az IceField osztály a players attribútumában tárolja, tehát ismerik azokat.

3.4.5 Player's Round

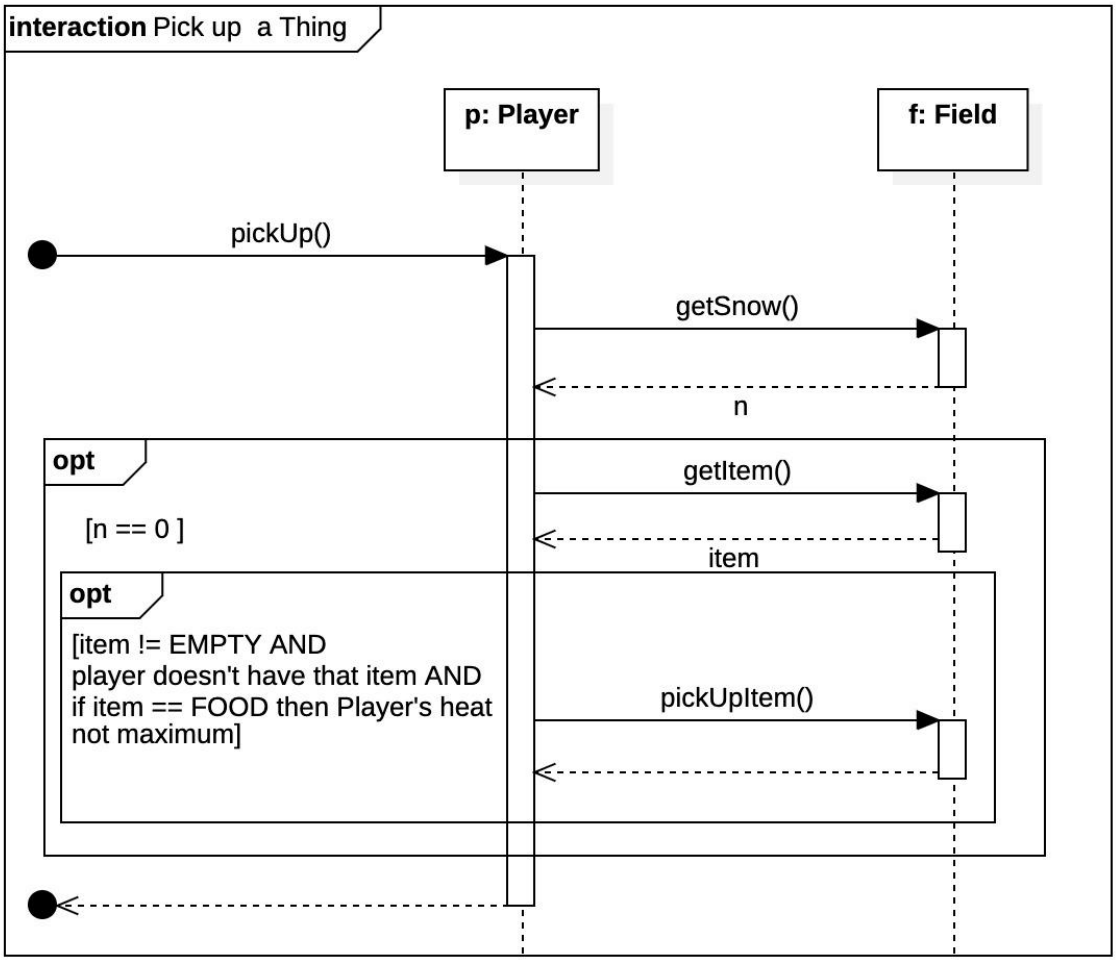


3.4.6 Clean the Field



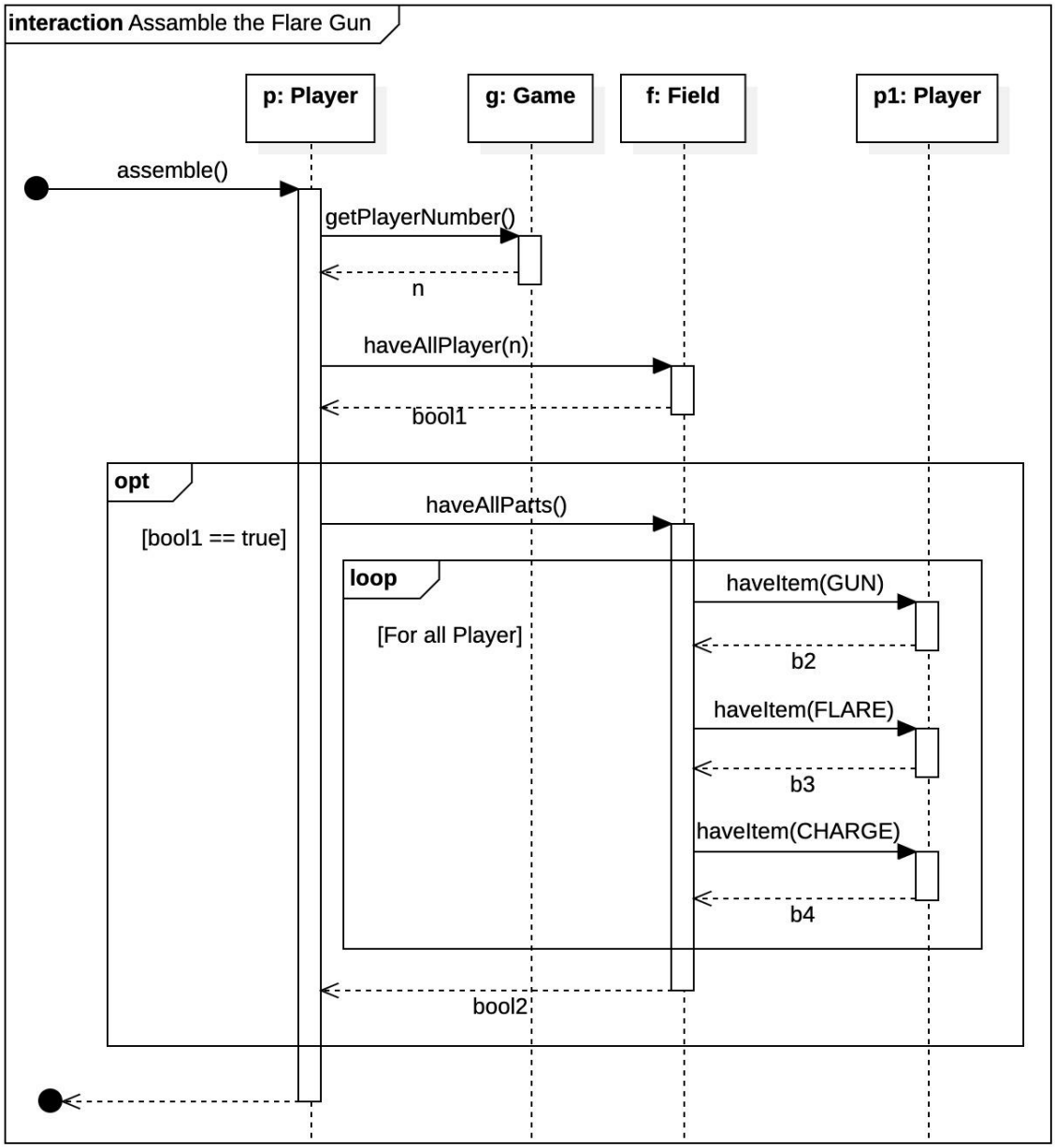
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.

3.4.7 Pick up a Thing



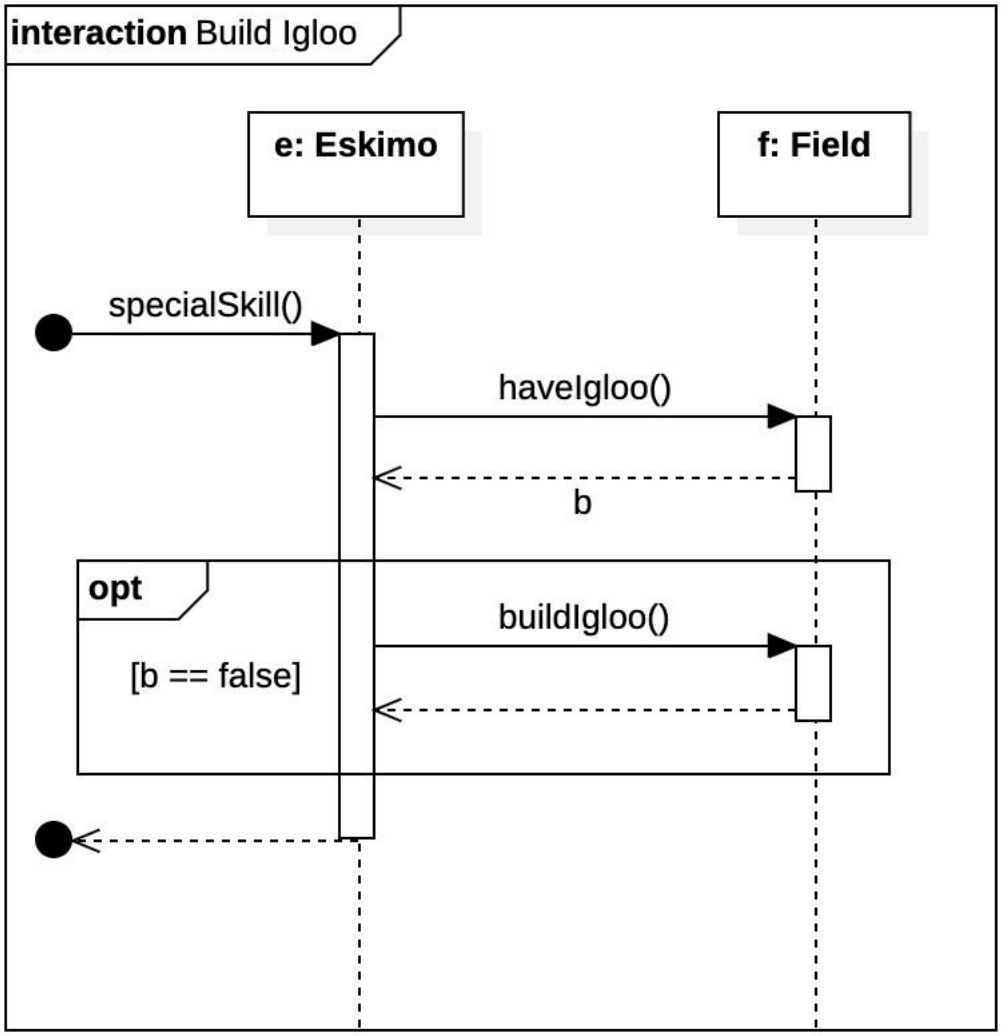
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.

3.4.8 Assamble the Flare Gun



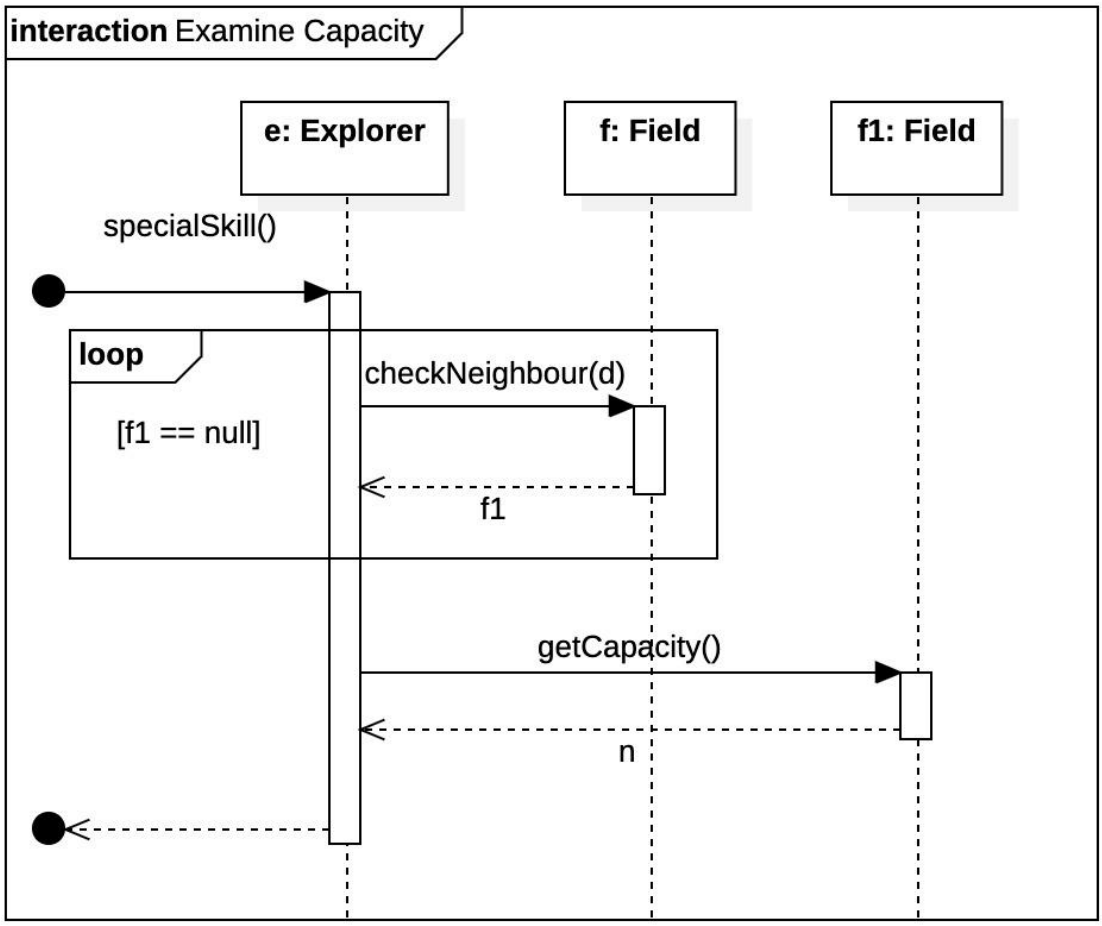
A g példány megegyezik a Player osztály game attribútumában tárolt példánnyal, tehát ismerik egymást.
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.
A Player példányokat a Field osztály a players attribútumában tárolja, tehát ismerik azokat.

3.4.9 Build Igloo



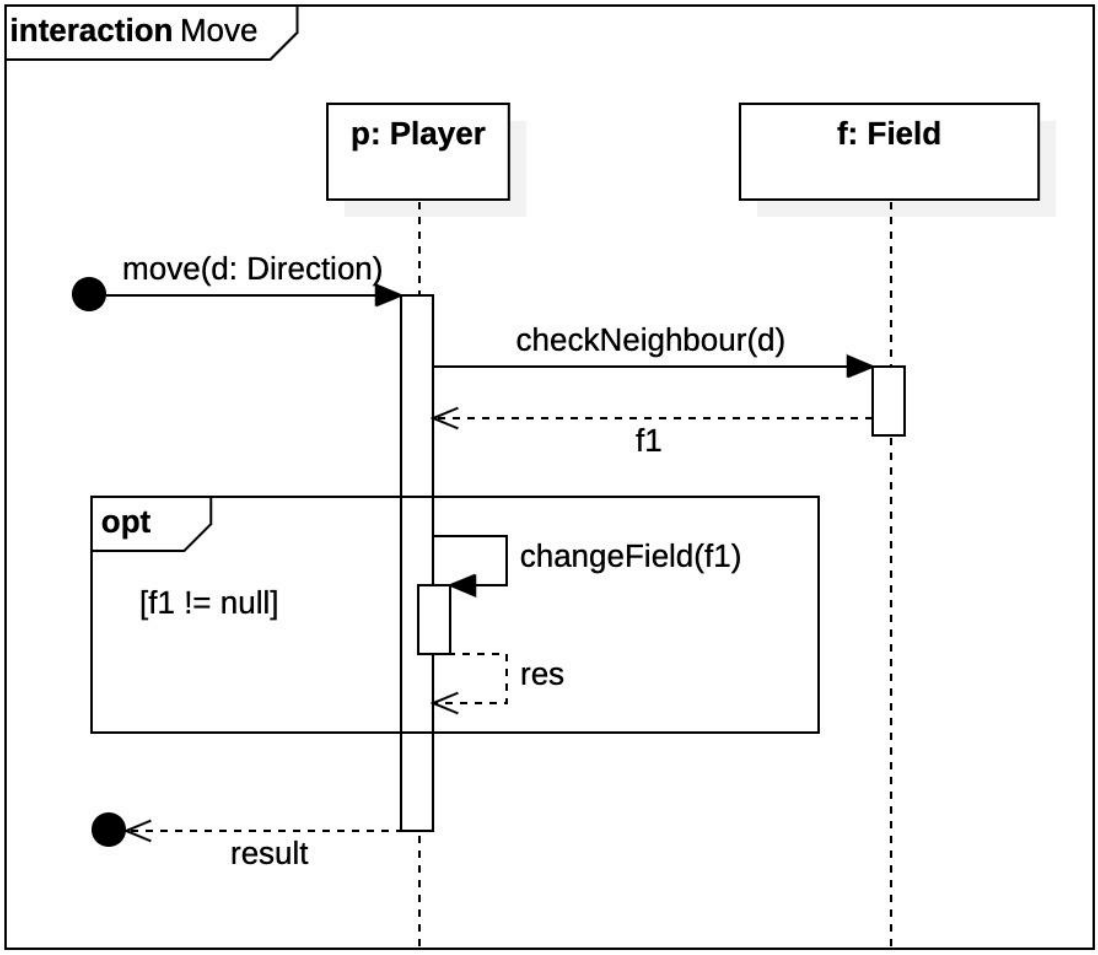
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.

3.4.10 Examine Capacity



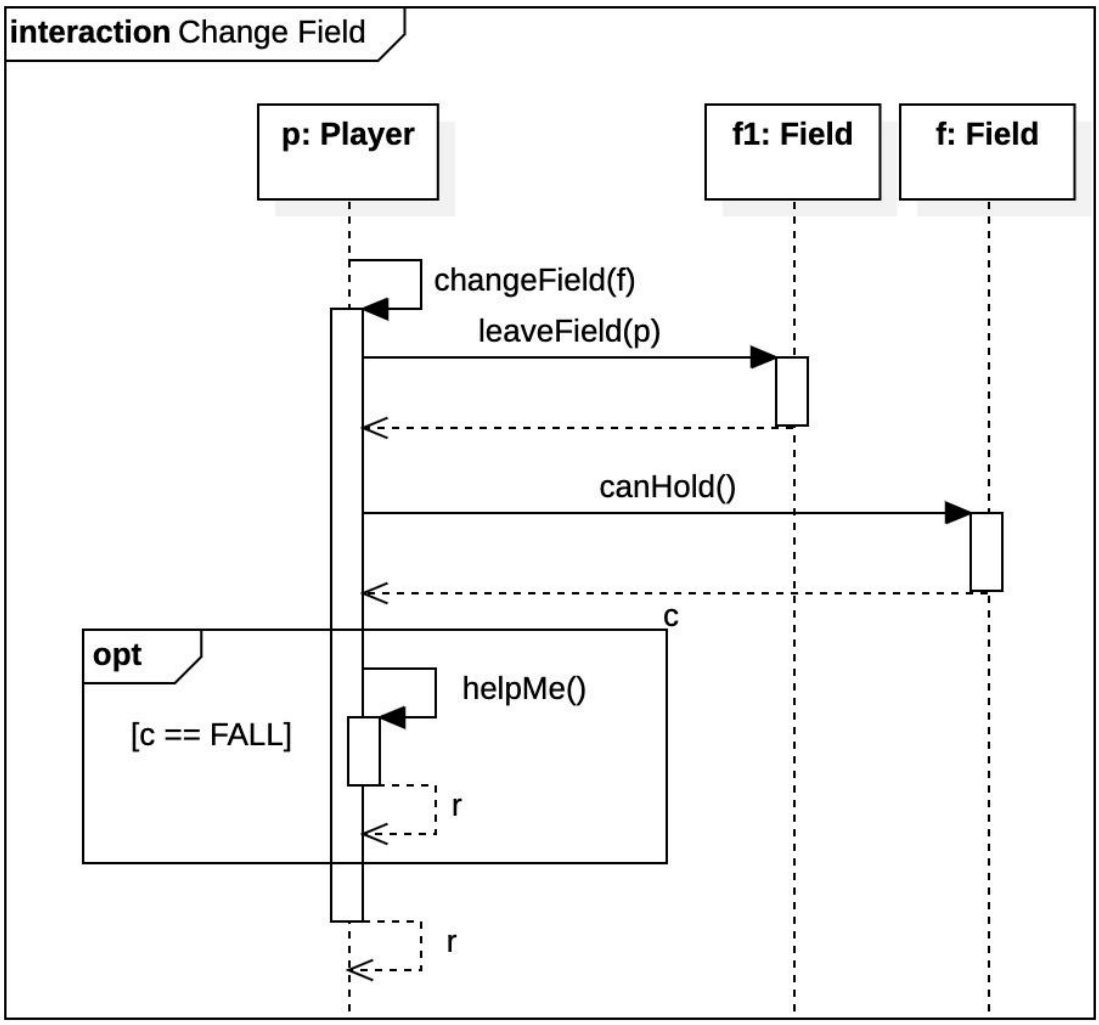
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.
A checkNeighbour függvény pedig visszatér az f2 referenciájával, tehát azt is ismeri a Player osztály

3.4.11 Move



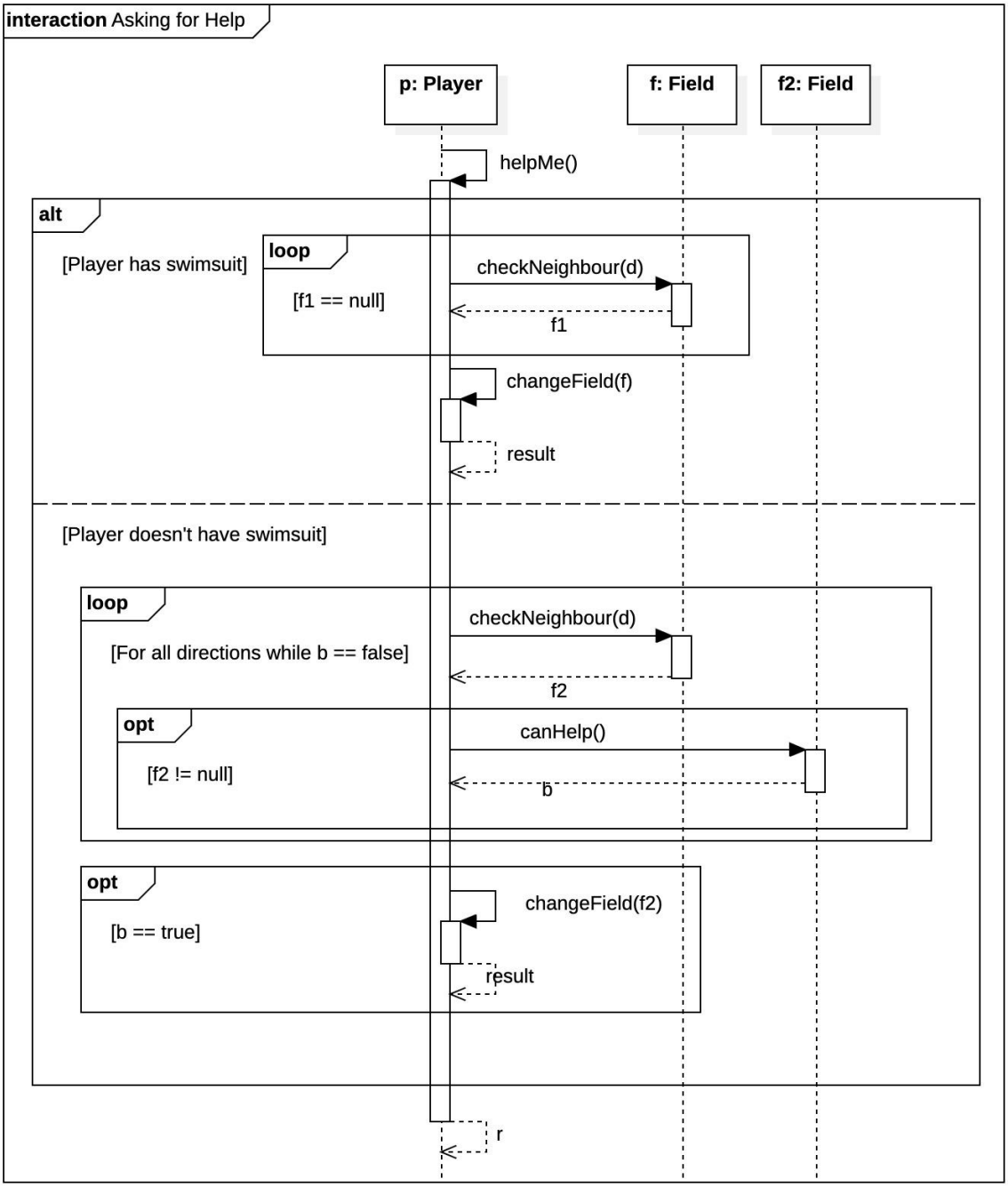
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.

3.4.12 Change Field



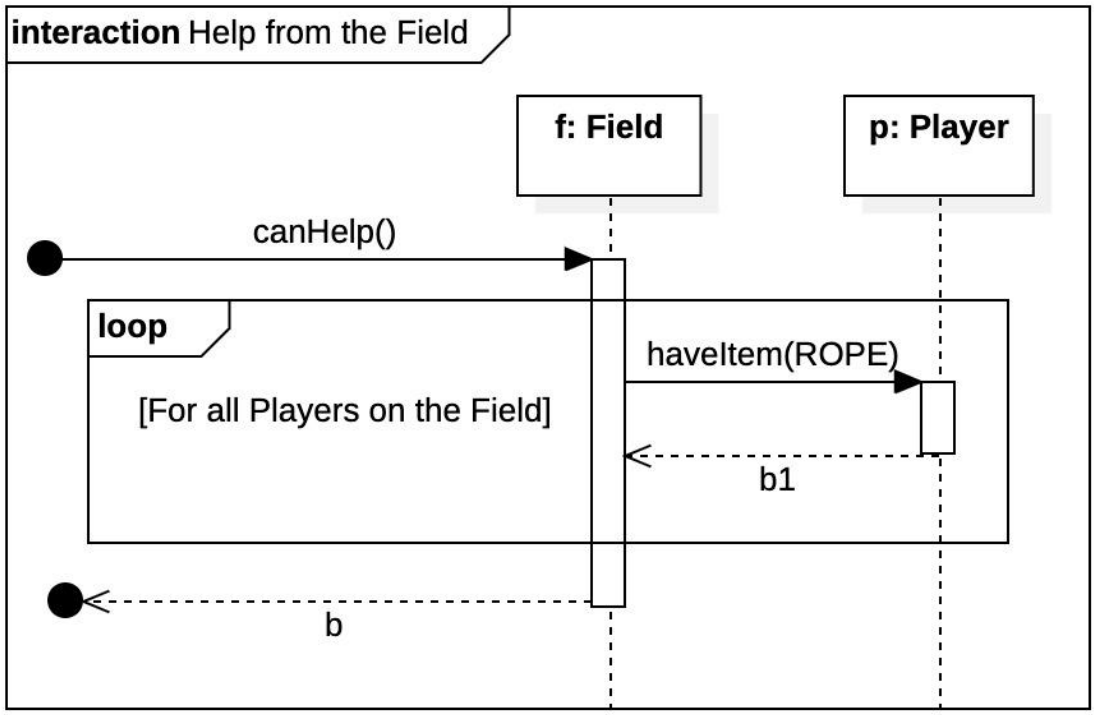
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást. (Pontosabban az actualfield attribútumba a changeField által átadott referencia kerül beállításra.)

3.4.13 Asking for Help



Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismerik egymást.
A changField függvény attribútumként azt a Field referenciát kapja meg, amelytől a canHelp TRUE értékkel tért vissza

3.4.14 Help from the Field



A Player példányokat a Field osztály a players attribútumában tárolja, tehát ismerik azokat.

3.5 State-chartok

Egyik osztály sem tartalmaz olyan állapotokat, melyek szemléltetése célszerű lenne state-charttal.

3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2020. 02. 25. 15:00	1,5 óra	Fábián	Osztálydiagram (3.2) vázának elkészítése
2020.02.27. 11:00	3 óra	Fábián Gutai Ilosvay Szabo K. Szabo M.	Értekezlet. Döntés: Osztálydiagram vázának egyeztetése, illetve feladatok felosztása
2020.02.28. 12:00	2,5 óra	Fábián	Osztálydiagram (3.2) elkészítése, rajzolása
2020.02.28. 14:30	1 óra	Ilosvay	Osztálydiagram (3.2) ellenőrzése
2020.02.28. 14:30	2 óra	Fábián Ilosvay	Javítási lehetőségek pontos átbeszélése és a lehetséges cselekvések miként jének megvitatása
2020.02.28. 16:30	0,5 óra	Fábián	Osztálydiagram (3.2) javítása
2020.02.28. 14:30	2 óra	Ilosvay	Osztálydiagram Field, Game és GameBoard osztályának dokumentálása (3.3.5-3.3.7)
2020.02.29. 10:00	3 óra	Fábián	Szekvencia diagramok (3.4) első harmadának elkészítése
2020.02.29 11:00	1 óra	Ilosvay	Osztálydiagram Hole és IceField osztályának dokumentálása (3.3.8, 3.3.9)
2020.02.29. 13:30	5 óra	Ilosvay	Osztálydiagram Chance, Direction, Eskimo, Explorer, Item, Player és Result osztályok dokumentálása (3.3.1-3.3.4 és 3.3.10-3.3.12)
2020.02.29. 14:00	3 óra	Fábián	Szekvencia diagramok (3.4) második harmadának elkészítése
2020.02.29. 20:30	2 óra	Ilosvay	Osztálydiagram dokumentálásának pontosítása (3.3)
2020.02.29. 20:00	3 óra	Fábián	Szekvencia diagramok (3.4) utolsó harmadának elkészítése
2020.02.29. 23:30	2 óra	Ilosvay	Osztálydiagram dokumentálásának pontosítása (3.3), szekvencia diagramok ellenőrzése
2020.03.01. 9:00	3 óra	Fábián	Szekvencia diagramok (3.4) ellenőrzése, javítása, illetve osztálydiagram (3.2) véglegesítése
2020.03.01. 13:00	1 óra	Fábián Ilosvay	Javítási lehetőségek pontos átbeszélése és a lehetséges cselekvések miként jének megvitatása
2020.03.01. 14:00	1 óra	Ilosvay	Osztálydiagram dokumentálásának javítása (3.3), leírások konzisztenssé tételének elkezdése a szekvencia diagramokkal
2020.03.01. 15:00	3 óra	Fábián	Osztálydiagram dokumentálásának javítása (3.3), leírások konzisztenssé tétele a szekvencia diagramokkal, és mindezek véglegesítése
2020.03.01. 15:20	2 óra	Szabó M.	Objektum katalógus (3.1) kidolgozása.
2020.03.02. 9:45	0,75 óra	Fábián	Objektum katalógus (3.1) javítása, kiegészítése.
2020.03.02. 11:45	1 óra	Szabó K	végső javítások, formázás